Descentralizando Sistemas de Telefonia IP utilizando o algoritmo Kademlia

Felipe de Castro Louback Rocha e Linnyer Beatriz Ruiz

***Resumo*— A maioria dos sistemas de Voz sobre IP (VoIP) são baseados em uma arquitetura cliente-servidor utilizando o protocolo SIP. Muitas pessoas e empresas se beneficiariam se os mesmos não exigissem a configuração e manutenção de servidores centralizados. Algumas propostas de descentralização são encontradas na literatura e, baseados nestas, propomos neste artigo uma solução baseada no algoritmo P2P Kademlia, visando melhorar o tempo de estabelecimento da chamada em tais ambientes.**

***Palavras-Chave*— Voz sobre IP, P2P, Descentralização, Rede de Computadores, Sistemas de Comunicação.**

***Abstract*— Most of Voice over IP(VoIP) systems are based in a client-server architecture using the SIP protocol. A lot of people and companies could be benefited if the system did not require the configuration and maintenance of central servers. Some proposals are found in the literature, and based on these we suggest a new solution based on the P2P algorithm Kademlia, with the intuition to improve call establishment time.**

***Keywords*— Voice over IP, P2P, Descentralization, Computer Network, Comunication Systems.**

I. INTRODUÇÃO

Sistemas de Voz sobre IP(VoIP - *Voice Over IP*) tem-se mostrado cada vez mais em uso nos dias de hoje, com diversos serviços sendo oferecidos no mercado. Prover serviços baseados em telefonia IP tem grande vantagens, entre elas redução de custos e maior mobilidade para os usuários. A configuração e manutenção de um sistema destes requer a configuração de servidores e requer mão de obra especializada, que entenda dos protocolos de sinalização utilizados para VoIP. Muitas empresas de pequeno e médio porte muitas vezes se vêem impedidas de fazer uso de uma infra-estrutura de telefonia IP pela falta de pessoal qualificado para manter e instalar um sistema e mesmo por não terem condições financeiras de contratar uma empresa terceirizada para fazer a instalação e manutenção do ambiente. Aplicações distribuídas e baseadas em uma arquitetura P2P(*Peer-to-Peer*) são cada vez mais comuns nos dias de hoje e com isto nota-se uma maior utilização das tabelas hash distribuídas(DHT- *Distributed Hash Table*). DHTs são conhecidas por terem características marcantes como descentralização, tolerância a falha e escalabilidade. O uso de DHT no lugar de servidores centralizados mostra-se uma alternativa atraente em diversos ambientes, inclusive nos ambientes de telefonia IP. Utilizando-se destas características das DHTs, propomos neste artigo mais uma abordagem de substituição dos servidores centralizados por DHTs, visando melhorar a eficiência da localização de recursos das abordagens existentes na literatura através de utilização de um novo algoritmo, o Kademlia.

Felipe de Castro Louback Rocha e Linnyer Beatriz Ruiz¸ Programa de Pós Graduação em Engenharia Elétrica, Escola de Engenharia, Universidade Federal de Minas Gerais, Belo Horizonte, Brasil, E-mails: louback@dcc.ufmg.br, linnyer@cpdee.ufmg.br.

A. Redes P2P

Uma grande variedade de definições existem para sistemas P2P(*Peer-to-Peer*). No nível mais básico, um sistema P2P é um sistema onde múltiplas aplicações de softwares interagem diretamente umas com as outras para realizar uma tarefa. O grupo de nós como um todo é muitas vezes referenciado como uma rede de sobreposição[1]. Isto contrasta com o tradicional modelo cliente-servidor, onde um pedaço de software centralizado (o servidor) processa requisições de diversos clientes. A escolha do modelo P2P ou cliente-servidor é uma decisão de arquitetura sobre onde o processamento da informação acontece.

Uma arquitetura P2P não necessariamente implica que cada nó deva fornecer todos os serviços ou nem armazenar todos os dados disponíveis. Coletivamente, todos os nós na rede de sobreposição devem prover todos os serviços, mas qualquer nó em particular pode prover apenas uma fração destes. Por exemplo, um conjunto de nós replicando uma banco de dados pode cada um, individualmente, armazenar um pequeno número de entradas do banco. Se um grupo muito grande de nós dividir a tarefa, as chances de uma determinada entrada estar em um dado nó é relativamente baixa, mas ao menos um nó na rede de sobreposição armazenará a dada entrada, garantindo que, como um grupo, os nós fornecem o serviço de banco de dados por completo.

Em uma arquitetura P2P estruturada os nós estão conectados uns aos outros em uma estrutura lógica e definida, como por exemplo um anel, uma árvore ou um *grid*. Muitos arranjos são possíveis, mas em geral é atribuído um identificador único para o nó quando o mesmo entra na rede de sobreposição. Este identificador do nó, pode ser atribuído de modo randômico, ou mais comumente, determinado pelo *hash* de alguma propriedade do nó, como seu endereço IP. O identificador do nó determina com quais outros nós o nó deve estabelecer contato. Por exemplo, um novo nó deve conectar-se a outros nós com identificadores ”próximos”ao seu, para alguma noção matemática de proximidade.

Um tipo especial de sistema P2P estruturado amplamente utilizado é a DHT(*Distributed Hash Table*). Alguns dos algoritmos de DHT mais discutidos hoje são Chord[1], Kademlia[7] e Bamboo[3]. Em uma DHT, não apenas a

[1]Do inglês *Overlay Network*

estrutura de conectividade entre os nós é controlada de forma matemática, mas também a maneira como os recursos são distribuídos entre eles. A cada recurso é atribuído um identificador, no mesmo espaço de identificadores que os identificadores dos nós. Isto é, a faixa de valores que o identificador do recurso e o identificador do nó podem assumir é a mesma. O identificador do recurso é o *hash* de alguma propriedade do recurso, como o nome do arquivo ou uma palavra chave que identifique o recurso. É feito o *hash* da palavra chave do recurso, produzindo o identificador do recurso e o nó com o identificador mais ”próximo”, armazena o recurso. A definição de *proximidade* e redundância são características dependentes do algoritmo utilizado.

B. SIP - Session Initiation Protocol

SIP [12] é um protocolo baseado em texto e derivado do HTTP(*Hiper Text Transfer Protocol*), portanto seus tipos de mensagens e tráfego são similares ao tráfego Web. SIP é um protocolo geral para estabelecimento e controle de sessões multimídia, mas sua maior utilização tem sido na área de VoIP.

Equipamentos que possuem SIP implementado podem ser configurados para comunicarem-se diretamente um com o outro, mas geralmente em sistemas com mais do que dois UAs[2](*User Agents*), um servidor central ou proxy é geralmente utilizado. Na especificação do SIP [12], as funções de um servidor são divididas, incluindo a função de servidor proxy e a função de servidor *registrar*(servidor de registro), que muitas vezes são implementadas juntas. Nos referiremos a estes servidores neste artigo simplesmente como proxy por questão de conveniência.

Um SIP *proxy* é um elemento de rede mais sofisticado do que um *proxy* web (*proxy* de páginas HTTP). Um SIP *proxy* mantém informação de localização sobre UAs e usuários, e também é responsável por todo roteamento e encaminhamento da sinalização entre os UAs.

C. OpenDHT

Sistemas distribuídos de grande escala são difíceis de serem empregados e tabelas hash distribuídas (DHTs) não são exceção a esta regra. DHTs aumentam a capacidade e disponibilidade das tabelas hash particionando o espaço de chaves entre um grupo de nós e replicando os dados armazenados, com isto conferindo características de descentralização, tolerância a falhas e escalabilidade.

Visando atender aplicações que fariam uso de uma DHT compartilhada, foi criada a OpenDHT[2]. OpenDHT é uma DHT compartilhada que roda em cerca de 200 máquinas do PlanetLab[5] globalmente distribuídas. OpenDHT implementa o algoritmo BambooDHT[3]. Nós que possuem o código da OpenDHT são considerados nós participantes da infra-estrutura. Clientes executam códigos que fazem uso da OpenDHT através de chamadas RPC(*Remote Procedure Call*), mas não possuem o código da OpenDHT implementado. Deste modo aplicações não precisam se preocupar com a implementação de uma DHT. Mas em contrapartida, nós não podem executar códigos específicos nos nós de infra-estrutura. Na maioria das aplicações de hoje, a DHT é executada através de uma chamada a uma biblioteca que a implementa. A adoção de uma biblioteca que implementa a DHT permite o uso de código especifico e otimizado para a aplicação, mas cada aplicação deve implementar sua própria DHT. OpenDHT oferece uma abordagem contrária a esta, com menor flexibilidade de utilização, mas em compensação não há a necessidade do programador da aplicação se preocupar com a implementação da DHT.

[2]*User Agents* mais comuns são telefones IPs em *hardware*, chamados de *hardphones* e em *software*, conhecidos como *softphones*.

D. Kademlia

Kademlia[7] é um sistema de tabela *hash* distribuída P2P. Kademlia utiliza a mesma abordagem geral que outras DHTs. Um ID(identificador) de 160 bits é associado a cada nó e é provido um algoritmo de procura que localiza sucessivamente nós mais próximos a um desejado ID, convergindo a procura ao objeto de forma logarítimica.

Kademlia trata os nós como folhas em uma árvore binária, onde a posição de cada nó é determinado pelo menor prefixo único do seu ID. Na figura 1, o nó preto não opaco mostra a posição de um nó com o prefixo único 0011 em uma árvore binária de exemplo.

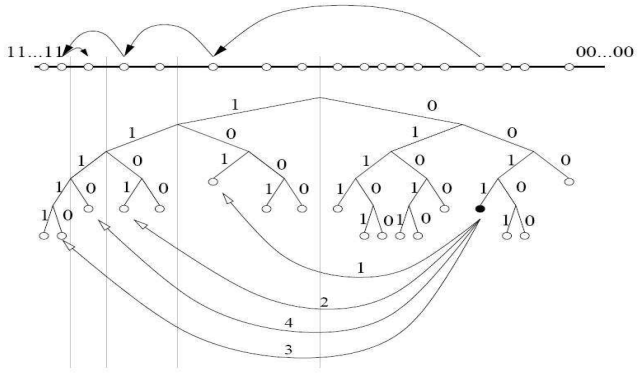

Fig. 1. Exemplo de um nó na árvore Kademlia e de uma procura por um determinado nó.

Para um determinado nó qualquer, a árvore é dividia em uma séria sucessiva de subárvores menores que não contenham o nó. A maior subárvore consiste da metade da árvore binária que não contém o nó. A próxima subárvore consiste da metade da árvore restante não contendo o nó e assim por diante. No exemplo do nó 0011, as subárvore consistem dos nós com prefixos 1, 01, 000 e 0010, respectivamente.

O protocolo Kademlia assegura que cada nó sabe de ao menos um nó em cada uma das suas subárvores, se aquela subárvore contiver um nó. Com esta garantia, qualquer nó pode localizar outro nó através de seu ID. A figura 1 mostra um exemplo do nó 0011 localizando o nó 1110 por meio de sucessivas perguntas ao melhor nó que ele conhece, tentando localizar o contato em subárvores cada vez menores, e finalmente a procura converge para o nó alvo. Geralmente as procuras são feitas de formas concorrentes, ou seja, mais de uma requisição é enviada ao mesmo tempo. O parâmetro α

controla este comportamento. No caso da figura 1 temos $\alpha = 1$ pois apenas um nó é questionado por vez, mas geralmente α assume valores maiores.

Cada nó Kademlia possui um ID de 160 bits. ID de nós são atualmente números randômicos de 160 bits. Chaves também são identificadores de 160 bits. Para atribuir um par <chave,valor> a um nó em particular, Kademlia se baseia em uma noção de distância entre dois identificadores. Dados dois identificadores de 160 bits, *x* e *y*, Kademlia define a distância entre eles como o seu ou exclusivo bit a bit(XOR), valor interpretado como um inteiro.

Nós Kademlia armazenam informações de contato uns dos outros para tornar possível o roteamento de mensagens. Para cada $0 < i < 160$, cada nó mantém uma lista de triplas <Endereço IP, Porta UDP, ID do nó> para nós de distâncias 2^i e 2^{i+1} de si mesmo. Estas listas são chamadas de k-buckets.

Maiores detalhes sobre o algoritmo podem ser obtidos em [7]. Kademlia é o algoritmo implementado em um dos softwares de compartilhamento de arquivos de maior popularidade nos dias atuais, o Emule.

II. Trabalhos Relacionados

Kundan Singh e Henning Schulzrinne da Universidade de Columbia em [8] propõem a utilização do OpenDHT como alternativa aos servidores de localização da arquitetura SIP.

SOSIMPLE [6] é um projeto que visa implementar um sistema inteiramente sem servidores, baseado em uma comunicação SIP P2P. Neste projeto o tráfego P2P é inteiramente transportado utilizando-se pacotes SIP e a DHT implementada é baseada no algoritmo Chord.

Skype [10] permite que usuários façam chamadas gratuitas para outros usuários Skype assim com ligações pagas para a rede pública de telefonia. Skype é parcialmente P2P, uma vez que conta com servidores centrais para autenticação e autorização. Skype é baseado em um protocolo proprietário e não é compatível com outros padrões como SIP e H323.

Kundan Singh e Henning Schulzrinne da Universidade de Columbia possuem um projeto [4] com uma abordagem bem semelhante ao do SOSIMPLE, utilizando o Chord como implementação da DHT. As principais diferenças dizem respeito ao transporte e roteamento dos dados da rede P2P.

Nossa abordagem se difere das demais por usar um algoritmo de DHT que suporta procuras concorrentes. Procuras concorrentes diminuem o tempo de resposta de localização um determinado recurso quando nós vivenciam diferentes condições de rede.

III. Solução Proposta

Nossa abordagem visa eliminar a necessidade do serviço centralizado de localização de recursos no SIP com a utilização de um algoritmo ainda não utilizado na literatura para tal fim. Eliminar o elemento centralizador significa eliminar a presença do servidor *proxy*. Em seu lugar será utilizado um adaptador que será responsável pela função de localização de recursos do *proxy*. O adaptador roda localmente na máquina em que o UA(*User Agent*) se encontra executando.

É requisito que não seja necessário alterar os UAs existentes, de modo que qualquer UA que implemente o protocolo SIP possa utilizar o adaptador. Visando atender a este requisito, optou-se por escrever um adaptador que roda localmente na máquina ao invés de se alterar um UA em específico.

Visando medir a performance da solução proposta, implementou-se duas versões do adaptador. Uma utilizando um serviço de DHT público, o OpenDHT, como alternativa ao servidor centralizado para armazenagem e localização de usuários, baseado no trabalho[8], e uma segunda implementação utilizando uma DHT implementada como biblioteca, o Kademlia, no próprio adaptador e que constitui a solução proposta pelos autores.

O adaptador foi escrito em C++, utilizando a biblioteca Resiprocate[11] para tratamento das mensagens SIP. A implementação do adaptador utilizando a OpenDHT foi feita com pequenos programas em Python, disponibilizados no próprio site do projeto OpenDHT, interfaceando com o programa em C++ para para permitir a comunicação entre o adaptador e o OpenDHT.

A implementação do adaptador que utiliza o Kademlia foi feita através de uma adaptação do código do projeto Anulus[9]. O projeto Anulus implementa o algoritmo Kademlia em C++ e um dos autores deste artigo faz parte da equipe de desenvolvimento do mesmo. O padrão SIP foi o escolhido devido ao mesmo ter se despontado como padrão para os fabricantes de equipamentos SIP e para provedores de serviços baseados em VoIP.

A. Comportamento do Adaptador

O processo de encaminhamento das chamadas pode ser feito adotando-se a abordagem de servidor de redireção (*redirection server*) ou de servidor *proxy*.

Quando um servidor se comporta como servidor de redireção, a ponta chamadora SIP envia uma mensagem `SIP INVITE` para o servidor, indicando a intenção de iniciar uma sessão com algum outro cliente(ponta chamada). O servidor de redireção então localiza o endereço IP da ponta a ser chamada. De posse do IP, o servidor envia uma mensagem `302 Moved Temporarily` para a ponta chamadora com o endereço da ponta a ser chamada. A ponta chamadora então envia um novo `SIP INVITE` direto para o IP indicado pela mensagem de redireção. Todo o fluxo de mensagens sequentes é feito diretamente entre a ponta chamadora e a ponta chamada. Quando o servidor se comporta como um *proxy*, praticamente toda a sinalização é intermediada pelo mesmo.

Foi-se adotado o comportamento de servidor de redireção para o adaptador. O servidor de redireção não precisa manter o estado das sessões estabelecidas, tornando sua implementação mais simples.

B. O Adaptador

O adaptador é responsável por realizar toda a tarefa de localização de recurso(usuários) e manutenção de usuários. Basicamente o adaptador pode ser dividido entre 2 módulos

principais: o módulo SIP, que implementa as funções de servidor de redireção e tratamento das mensagens SIP, e módulo P2P, que implementa toda a manutenção e localização de usuáros na rede P2P. A comunicação entre estes dois módulos é feita através de uma interface exportada pelo módulo P2P permitindo a localização e registro de usuários.

No caso da implementação com o OpenDHT, o módulo P2P é substituido por um módulo mais simples, responsável por fazer a inserção e procura de chaves no OpenDHT.

C. Processo de Inicialização

Para entrar na rede P2P, um nó precisa encontrar um outro nó que faça parte da rede de sobreposição. Para termos uma referência de algum nó que se encontra na rede, utilizamos o OpenDHT para armazenar tal referência. A idéia de armazenar a referencia no OpenDHT é baseada na idéia de DNS dinâmico, mas utilizando uma DHT pública compartilhada para tal função. No DNS dinâmico as entradas são atualizadas em tempo real. Esta é uma pratica comum em usuários de ADSL, que a cada conexão tem seu IP alterado por causa do DHCP da provedora de Internet.

A referência é armazenada utilizando como **chave** `bootstrap.node.br` e como **valor** associado à chave *IP:Porta* de um nó da rede P2P. Deste modo, quando um nó fizer uma procura pela chave será retornado o IP de um nó na rede P2P.

Quando o adaptador é inicializado, o mesmo faz uma procura pela chave `bootstrap.node.br` no OpenDHT. No caso de sucesso, o nó inicializa o processo de *bootstrap*[3] com o nó retornado. Quando a procura não retorna nenhum IP, inicialmente o nó tenta registrar no OpenDHT a chave `bootstrap.node.br` com o seu IP. Assim ele se torna referência para nós futuros que entrarem na rede. Após a tentativa de registro no OpenDHT, o nó tenta localizar algum outro nó na rede P2P através do envio de um *broadcast* na rede local. *Broadcast* é utilizado pois a falha na localização de um registro no OpenDHT pode ser indicador de alguns fatores, dentre eles ausência de comunicação com o OpenDHT causado por queda/perda na comunicação com a Internet; ou que deseja-se estabelecer um sistema de Telefonia IP em uma rede que não tenha conectividade com a Internet; ou mesmo que este nó é o primeiro nó na rede P2P.

Se algum nó estiver presente na rede P2P e notar o *broadcast* de procura por outros nós, ele responde com o seu IP. O nó que originou a requisição então inicializa o processo de *bootstrap* com o nó que respondeu. Apenas a primeira resposta é considerada.

O adaptador implementado para utilizar o OpenDHT não precisa deste processo, pois sempre um nó no OpenDHT é encontrado pelos programas em Python fornecidos pelo próprio site do projeto. Na ausência de conectividade com a Internet, os nós do OpenDHT não serão encontrados.

D. Registro de Usuários

Ao receber uma requisição de registro, o adaptador realiza a inserção do registro do usuário na DHT, sendo ela o Kademlia ou o OpenDHT. A idéia do processo é a mesma.

Durante o processo de registro, o UA envia uma mensagem `SIP REGISTER` para o adaptador. O módulo SIP percebe que o pacote é um pedido de registro. As DHTs armazenam pares <chave, valor>. Ao localizar um usuário em um serviço de telefonia IP, o UA precisa obter as informações de IP e porta que um determinado usuário se encontra para que a chamada possa ser estabelecida. Então, ao utilizar uma DHT para armazenar as informações, armazenamos o par no formato <usuário, IP:Porta>, assim quando uma procura for feita por um usuário, será retornado o IP e a porta no qual o UA do mesmo se encontra.

E. Manutenção de Usuários

A principal aplicação do Kademlia atualmente é compartilhamento de arquivos. Ao se procurar por uma determinada chave, os nós que participam da procura, ou seja, estão no caminho da procura, armazenam uma cópia da resposta. Deste modo, procuras subsequentes a esta chave tendem a serem mais rápidas, pois a probabilidade de um nó estar com a chave em *cache* é maior.

No caso de telefonia IP, o registro de usuários não pode ser persistente pois os mesmos podem entrar e sair da rede a qualquer momento, podendo obter um IP diferente a cada entrada. O comportamento do algoritmo Kademlia precisou ser alterado para atender a necessidade do serviço de telefonia IP, sendo acrescentado um tempo de expiração de 3600 segundos à chave relativa ao usuário, após o qual a mesma é apagado da DHT.

Quando o adaptador recebe um pacote `SIP REGISTER`, uma nova chave é inserida na DHT e o tempo de expiração inicializado para 3600 segundos. Ao receber um novo pacote `SIP REGISTER`, uma nova inserção da chave é feita e o nó responsável por aquela chave atualiza o tempo de expiração novamente para 3600 segundos. O tempo entre o envio de mensagens `SIP REGISTER` geralmente é menor do que 3600 segundos.

Cópias em *cache* do registro do usuário não são atualizadas. Mas a expiração de cópias em *cache* não vem a ser um problema pois, como característica do algoritmo Kademlia, todas as procuras convergem para um mesmo nó. Então, no caso de todas as cópias em *cache* terem expirado, durante uma nova procura, novas cópias se formarão.

Para tratar a manutenção de usuários no OpenDHT, a abordagem adotada foi semelhante, sendo que a cada mensagem `SIP OPTIONS` recebida, uma nova inserção para a chave era feita, atualizando o seu tempo de expiração.

F. Localização de usuário e estabelecimento de Chamadas

Como dito anteriormente, nosso adaptador se comporta como um servidor de redireção. Quando o UA deseja estabelecer uma ligação com outro UA, uma mensagem `SIP INVITE` é enviada para o adaptador que então inicia o processo de localização do usuário na DHT. Caso seja encontrado, uma mensagem `302 Moved Temporarily` é montada, de acordo com a RFC 3261[12], contendo as informações

[3]Processo de inicio e registro na rede de sobreposição P2P através de troca de informações com algum nó que faça parte da rede.

necessárias para o re-encaminhamento da chamada. Esta mensagem é retornada para o UA e este envia o `SIP INVITE` diretamente ao UA da ponta a ser chamada. No caso em que a chave não é encontrada, uma mensagem `404 Not Found` é enviada de volta ao UA. Este comportamento pode ser visto na figura 2.

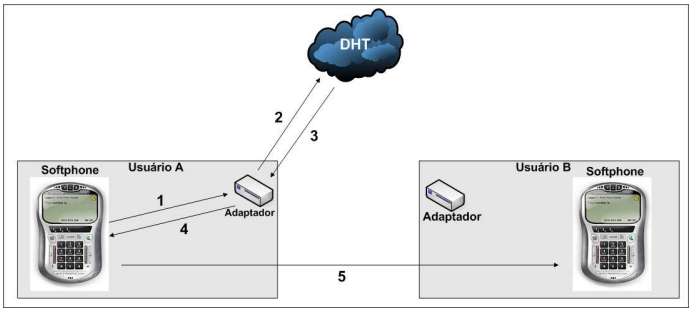

Fig. 2. Estabelecimento de Chamada com adaptador se comportando como servidor de redireção.

IV. Medições

Em uma arquitetura cliente-servidor, localização de usuários é feita em $O(1)$ enquanto que em DHTs são feitas em no máximo $O(log(n))$ [7], [1]. Este é um preço a se pagar pela descentralização. O tempo de localização de usuários; influência no tempo de estabelecimento da chamada. Um modo de se medir a eficiência de um sistema descentralizado é medir o tempo gasto na localização de usuários. Visamos medir este tempo de latência na solução não centralizada proposta em [8] e implementada pelos autores, assim como a solução proposta pelos autores.

Inicialmente registrou-se um usuário no OpenDHT e pesquisas por aquele usuário foram feitas durante 7 dias, no intervalo entre as 17:00hs e 18:00hs. Foram feitas 10 medidas por dia. As medidas encontram-se no gráfico da figura 3.

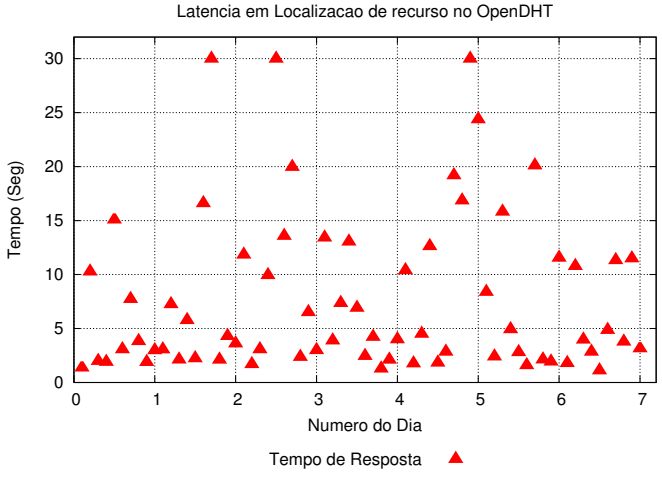

Fig. 3. Procuras feitas no OpenDHT ao longo de um dia

A utilização do algoritmo Kademlia visa diminuir o tempo de estabelecimento de uma chamada em ambientes no qual parte dos nós experimentam congestão na rede. Para medir este tempo em nossa solução, estabeleceu-se o seguinte ambiente de testes: foram inicializados 30 nós em uma rede local. Escolheu-se uma rede local pois os maiores beneficiados de nossa abordagem são pequenas empresas, ou cenários *ad-hoc*, que podem ser caracterizados e/ou simulados como uma rede local.

Inicialmente foram feitas 40 procuras aleatórias por usuário em um intervalo de 1 hora. Após isto, codificou-se metade dos nós para gerar um atraso de 2 segundos antes de enviar a resposta e realizamos os experimentos novamente. Então realizou-se o mesmo experimento mas codificando 4 segundos de atraso. A aplicação possui um tempo de retransmissão de 3 segundos codificado. Então estes cenários simulam ambientes no qual nenhuma congestão é vivenciada, congestão é vivenciada mas não gera retransmissão e congestão é vivenciada e é gerado retransmissão dos dados.

No algoritmo Kademlia, α é um parâmetro de concorrência nas procuras, conforme explicado na seção I-D. Visando comparar o comportamento da solução para buscas concorrentes e não concorrentes, fez-se testes para valores de $\alpha = 1$ e $\alpha = 5$. As abordagens [4], [6] são baseadas no algoritmo Chord[1], um algoritmo não concorrente($\alpha = 1$). Os resultados com $\alpha = 1$ podem ser vistos no gráfico da figura 4 e com $\alpha = 5$ na figura 5.

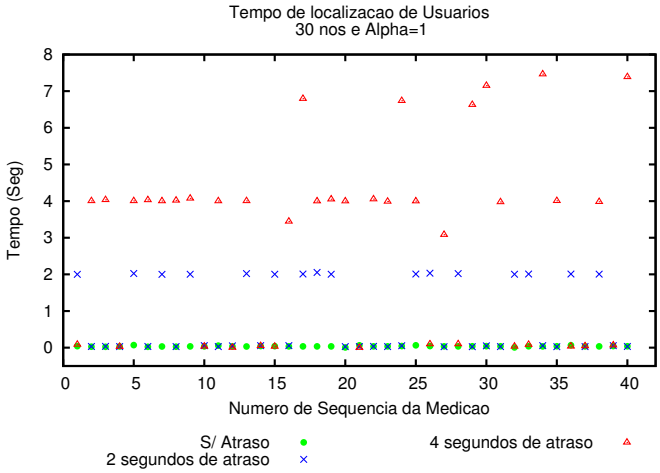

Fig. 4. Localização de recurso no Kademlia com $\alpha = 1$.

O OpenDHT apresentou tempos de resposta muito variáveis, apresentando em alguns momentos tempos reposta de mais de 30 segundos. Esta imprevisibilidade do tempo de resposta é prejudicial para sistemas de telefonia IP pois implicam em um tempo de estabelecimento de chamada elevado.

Para nossa abordagem, com $\alpha = 1$, temos um tempo médio de 0,38052, 0,828502 e 3,041073 segundos para ambiente sem atraso, com 2 segundos de atraso e 4 segundos de atraso simulado, respectivamente. Para a abordagem concorrente, com $\alpha = 5$, o tempo médio de resposta melhora consideravelmente. Também nota-se que não há uma diferença expressiva no tempo médio de reposta para o ambiente sem congestão simulada e para os ambientes com congestão simulada, obtendo-se um tempo médio de 0,038052, 0,061473 e 0,068874 segundos

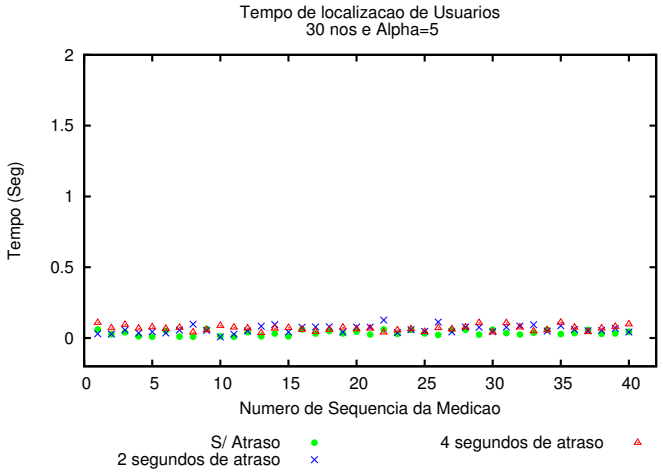

Fig. 5. Localização de recurso no Kademlia com $\alpha = 5$.

para os mesmos casos simulados com $\alpha = 1$.

Como a pesquisa é feita de forma concorrente, mesmo que uma requisição chegue a um nó que possui atraso simulado configurado, outro nó sem atraso simulado provavelmente receberá a requisição e responderá. Como são enviadas 5 requisições simultâneas, não será necessário esperar um *time-out* para o envio de outra. Com isto há uma redução no tempo médio de resposta, conforme mostra o gráfigo da figura 5.

V. Trabalhos Futuros

A. NAT Traversal

A questão de usuários poderem estar atrás de NAT é uma questão em aberto. No projeto atual, é assumido que todos os usuários se encontram em endereços roteáveis entre si. Uma alternativa seria a implementação de alguma solução como STUN(*Simple Traversal of UDP through NAT*) nos adaptadores que possuírem um IP válido na Internet.

B. Autenticação

No momento ainda não temos nenhum tipo de autenticação sendo provida e simplesmente assumimos que todos os usuários do sistema são confiáveis. Mas esta é uma suposição que não pode ser feita em ambientes reais. As soluções listadas na seção de trabalhos relacionados também não tratam a autenticação de forma efetiva. O Skype possui servidores centralizados, o que fere a nossa premissa de descentralização das funções do sistema. Os demais possuem sugestões de autenticação, mas nenhuma implementada, testada e adotada de fato. Um sistema de autenticação ainda é assunto para ser melhor estudado e avaliado.

VI. Conclusões

A abordagem atual do trabalho mostrou-se efetiva mostrando que é possível possuir uma abordagem eficiente de descentralização em sistemas de telefonia IP baseados no protocolo SIP utilizando o protocolo Kademlia. O adaptador proposto atualmente trabalha em conjunto com *softphones*, mas uma mesma abordagem poderia ser feita para se codificar um adaptador semelhante e embuti-lo em equipamentos como aparelhos telefônicos IP.

Nosso projeto difere-se dos demais por trazer uma solução híbrida envolvendo tanto uma DHT pública como uma própria implementada(Kademlia), sendo a pública utilizada como alternativa a um serviço de DNS dinâmico e a implementada como alternativa ao serviço de localização de recursos do SIP.

Outro diferencial é funcionamento da solução em ambientes aonde ocorra perda de conectividade ou falha de comunicação com a Internet, possibilitando que a mesma seja usada em ambientes *ad-hoc*. Pequenas/médias empresas com falta de mão de obra especializada são exemplos de ambientes que se beneficiariam com um sistema de Voz sobre IP auto gerenciável. Não é de conhecimento dos autores nenhum outro sistema que funcionem em ausência de conectividade com a Internet.

Nossa solução é a primeira solução de descentralização de serviços de telefonia IP que explora o algoritmo Kademlia. O fato do algoritmo utilizar procuras concorrentes mostra-se uma vantagem em relação as demais abordagens pois reduz significativamente o tempo de localização de usuário no caso de perda de pacotes afetarem diversos nós da rede. Até o presente momento, os autores não encontraram nenhum outro trabalho que explore este algoritmo para tal propósito, sendo este trabalho o primeiro.

O comportamento do adaptador como um servidor de redireção, eliminando a necessidade de se manter o estado das conexões pelo adaptador também é uma novidade em relação às demais abordagens existentes. O fato de o adaptador ter sido codificado para ser compatível com o protocolo SIP permite que o mesmo possa ser utilizado com qualquer softphone que seja compatível com a RFC do SIP [12].

Referências

[1] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek and Hari Balakrishnan, *Chord: A scalable peer-to-peer lookup service for internet applications*, SIGCOMM, San Diego, CA, USA. Agosto 1991.

[2] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I.Stoica, and H. Yu. *OpenDHT: a public DHT service and its uses*, In: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications. Philadelphia, Pennsylvania, USA, 2005.

[3] Bamboo. http://bamboo-dht.org/

[4] K. Singh and H. Schulzrinne, *Peer-to-Peer internet Telephony using SIP*, Columbia University Technical Report CUCS-044-04, New York, NY, Oct 2004.

[5] Planetlab. http://www.planetlab.com/

[6] Bryan, D.A., Lowekamp, B.B., Jennings, C, *SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System*, in: Advanced Architectures and Algorithms for Internet Delivery and Applications, 2005. AAA-IDEA 2005. First International Workshop on Advanced Computing, June 2005.

[7] P. Maymounkov, D. Mazieres. *Kademlia: A peer-to-peer informatic system based on the XOR metric*, in: Proc. of the 1st International Peer To Peer Systems Workshop, 2002.

[8] K. Singh and H. Schulzrinne, *Using an External DHT as a SIP Location Service*, Columbia University Technical Report CUCS-007-06, New York, NY, Feb 2006.

[9] Anulus Project. http://code.google.com/p/anulus/

[10] Skype. http://www.skype.org/

[11] Resiprocate. www.resiprocate.org/

[12] J. Rosenberg, Henning Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, (2002). *SIP: session initiation protocol - RFC3261*, Internet Engineering Task Force.